(1)

"Synchronization"

pc العادى بيستخدم clock puls (rising أو Falling)
سرعه as computer مرتبطه بسرعه المسارات الداخليه أى كل ما العمليه بين 2 edges تتم
اسرع بيكون الجهاز شغال أسرع

---

* المقصود به المجموعه التى تشتغل مع بعضها .

محتاجينه ليه - الزمن - زمن المجموعه - هنطبق على أفرادها ولا standard المحتوى .

---

why it needed ⇒

① clien to server أو peer to peer (احنا عن أحمد الأنواع) احنا كل واحد بيشغل
client ووضع آخر server
⇒ time , ack , TCP " msgs فى ترتيب زمنى بين "
المتبادله .

② scheduling taskes. → multiprocessors
دون مش بيمثل مشكله فى ان الجميع بيقرا ال resources ولكن المشكله فى حاله احدهم عدل ع data.
يلزم وقتها عمل check عليها .

③ adjusting time ≅ synchronization

فى حاله ان 2 client مختلفين فى الوقت حيث ممكن يتم عمل drop لل data فى حين
أنها متأخر فيظهر انه متأخر فى التغير "الأقدم" فى حين انه الأحدث .

المثالى ان تكون كل الأجهزه فى العالم انها تصبح synchronized .

يحتاج precedence order الترتيب الزمنى لترتيب ال Taskes الموجوده

Computer logical time :
كل computer يحتوى على كريستاله تستوى عدد من oscillators. تحدد الزمن
ولكن مع مرور الوقت قد تختلف .

②

كل oscillation يتعمل Decrement Counter وكل لما يصبح Counter = 0 يزود الزمن 1s
by 1

حيث يتعمل interrupt ويعمل reloading للعقبة عدد C = 1000 ويعاد العملية مره آخرى
وتطلع عليها clock Tick

---

1 ادخال ال Time في بداية Booting يعتبر مسئوليه user.

S.W clock تعتمد ع الوقت اللى احنا مدخلينه و Ticks اللى تأتى بعدها حيث كل واحدة دى بيزود 1sec.

---

clock skew ⟸ (±) الفرق في الوقت بين كل 2 client اى الأجهزه وبعضها "

وخلوها ←

① عدم تحديد وجود علاقة بين Time وبعضها.
"happen Before" مين حصل event قبل مين.

لو عندنا 2 Tasks a,b و a يتنفذ قبل b وعند توزيع Tasks لا يوجد تواصل معلومه
لكن تطول تدل ع انه a اتنفذ قبل b

Lamport logical clock

أى كل جهاز يشتغل ع ساعته عادى ولكن عند Communication بين الاجهزه
يجب عل Synchronize بينهم أى يظبطوا الوقت مع بعض
بحيث يصبح sequence متتابعا logical.

Time مش ينفع يكون بال sec.

$e < e' \Rightarrow C(e) < C(e')$
clock.

هناك تكرار حيث ان الفيزيكل هو المعمول 10 8 ↘9
ولكن فيه مشاكل.

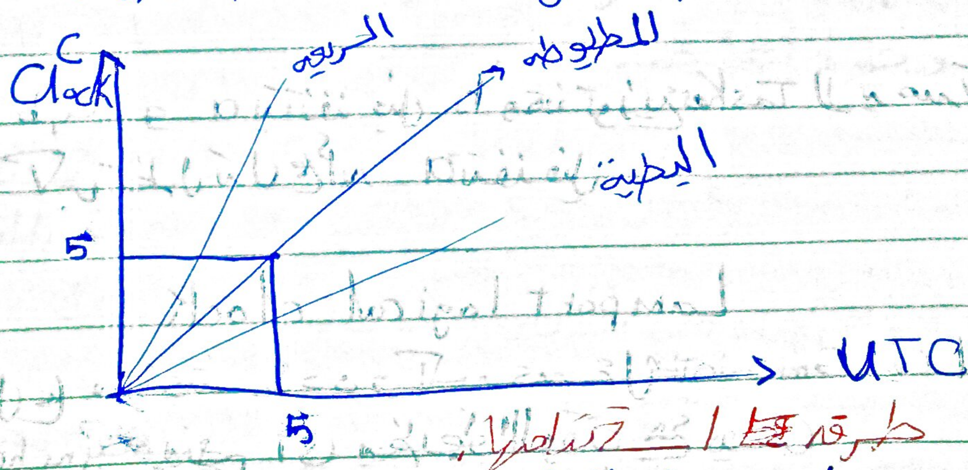
"physical clock syncronization"

حيث يتواجد هناك ساعة كونية مظبوطة رغم أي تغيرات.
• ولكن مع مرور الوقت قلت سرعة دوران الأرض حيث أصبحت تأخذ وقت أطول.
• ولذلك تم ضبطها عن طريق الشمس في مدار اليوم.

Atomic clock

• ساعة شمسية متواجدة في أماكن معينة "Universal coordinated time"
وكل فترة يتم تبطيئها 1sec "solar hour"

• مكانها متاح للناس ويتم عمل Broadcasting للعالم كله عن طريق الأقمار الصناعية.



طريقة اتحادها

① Time server هو العادي الذي time وبنظبط عليه.
(C)

"Algorithm"

١ Time server مرتبط UTC كل فترة بيعمل interrupt ويظبط نفسه مع
UTC اللي ماسكها عن طريق satalites أو عن طريق "website"
وبعدها بيجي client بيعمل access على server ليظبط نفسه.

كل oscillation بتعمل decrement Counter by 1 وكل ما يصبح Counter = 0 بيزود الزمن دا

حيث يعمل interrupt ويعمل reloading لعقه C = 1000 عدد ويكاد العمليه مره آخرى
وتلك الحاله clock Tick

1 دخال ال Time في بدايه Booting يعتبر مسئوليه user .

s.w clock تعتمد ع الوقت الي احنا مدخلينه و Ticks الي تأتي بعدها حيث كل واحده دي
بيزود 1sec.

clock skew ⟸ (±) الفرق في الوقت بين كل client مع الأجهزة وبعضها "
وحلوا

(1) عدم تأكيد وجود علاقه بين Time وبعضها .
" happen Before " مين حصل event قبل مين .

لو عندنا 2 Tasks a,b و a يتنفذ قبل b وعند توزيع Tasks لا يوجد تواصل معلومه
لكن تظل تدل ع أنه a تنفذ قبل b

Lamport logical clock

أي كل جهاز يشتغل ع ساعته عادي ولكن عند عمل Communication بين الأجهزه
يجب عمل Synchroniz يتم أي يظبطوا الوقت مع بعض
بحيث يصبح sequence بتاعتنا logical.

Time مش ينفع يكون بال .sec

$e < e' \Rightarrow c(e) < c(e')$

clock.

(4)

$\frac{T_1 - T_0}{2}$

T_0 → الزمن هنا → Time server

T_1

(1) RTT

Rounte Trip Time

example

To 10:00

T_1 10:03

10:02

$\text{round time} = \frac{T_1 - T_0}{2}$

زمن رايح او زمن الجاي

$= \frac{3}{2} = 1.5$

New Time of the machine $= \text{UTC Time} + \text{round Time}$
$= 10:02 + 00:01.5$
$= 10:03.5$

العيب انه بيتفرض بعض الافتراضات وقد تكون غير متحققة.
حيث اذ الطريق الذهاب نفس طريق العودة
والزمن الى اخذه الذهاب = زمن العودة
في حين وجود اي اختلاف

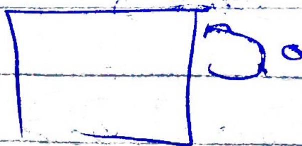
(2) UNIX algorithm

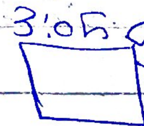
deamon شيطان
ولكن المقصود به أنه سريع جداً

هو هنا هو الى بينادي الأجهزة لكي تظبط نفسها.

يجمع الفروق وبعدها يقسم على عددهم يحسب ((avrge Time)) وبعدها يضيفه عليهم كلهم

$\frac{0 + 25 - 10 = 15}{3} = 5$



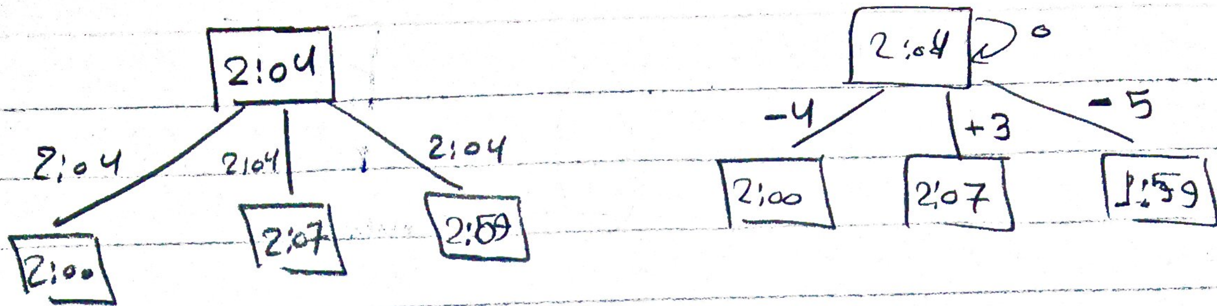


$-20 = 5 - 25$

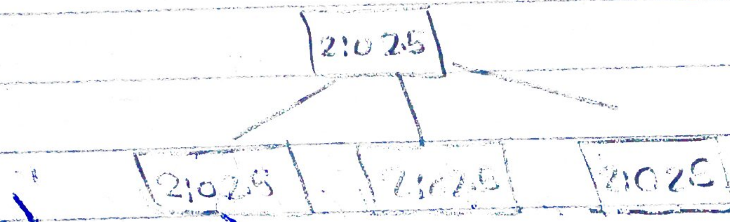
$5 - (-10) = 15$







$$\text{average error} = \frac{-4+3-5+0}{4} = -1.5$$



New time = time + average

"decentralized average algorithm"

كل فترة بيعمل عليه التظبيط ويصبح كل جهاز مسؤول عن نفسه حيث يعمل Broadcast ويشوف الفروق ويحسبها لنفسه ويظبط نفسه لوحده واللي عايز يعمل زيه مش عايز يظبط نفسه هو حر.

لذلك كده نكون شالت العيب من single machine

ينتظر period time والأجهزة اللي بتلحق ترد يقسم على عددهم فقط.
واللي مش عايز يرد هو حر.
" felixability المرونة "

في حالة وجود فروق كبيرة وصغيرة جدًا يتم تجاهلهم وحساب المتوسط على الباقيين.

round trip → ممكن يكون ثابت لجهازهم
أو يكون مختلف فيهم أيضًا ده